A PRELIMINARY REPORT ON

"GENETIC ALGORITHM"

SUBMITTED TO THE SAVITRIBAI PHULE PUNE UNIVERSITY, PUNE

IN THE PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE AWARD OF THE DEGREE

OF

BACHELOR OF ENGINEERING (COMPUTER ENGINEERING)

SUBMITTED BY

AJINKYA SONAWANE	**Exam No :B150394374**
RAGINI BETRABET	**Exam No :B150394222**

DEPARTMENT OF COMPUTER ENGINEERING

AIM-The aim of this project is to apply genetic algorithm for optimization on a dataset obtained from UCI ML repository For Example(TSP).

OBJECTIVE-The objective of this project is to understand the working of genetic algorithm.

THEORY-

GENETIC ALGORITHM

Genetic Algorithms(GAs) are adaptive heuristic search algorithms that belong to the larger part of evolutionary algorithms. Genetic algorithms are based on the ideas of natural selection and genetics. These are intelligent exploitation of random search provided with historical data to direct the search into the region of better performance in solution space. They are commonly used to generate high-quality solutions for optimization problems and search problems.

Genetic algorithms simulate the process of natural selection which means those species who can adapt to changes in their environment are able to survive and reproduce and go to the next generation. In simple words, they simulate “survival of the fittest” among individual of consecutive generation for solving a problem. Each generation consist of a population of individuals and each individual represents a point in search space and possible solution. Each individual is represented as a string of character/integer/float/bits. This string is analogous to the Chromosome.

Foundation of Genetic Algorithms:

Genetic algorithms are based on an analogy with genetic structure and behavior of chromosome of the population. Following is the foundation of GAs based on this analogy:–

1. The individual in population compete for resources and mate.

2. Those individuals who are successful (fittest) then mate to create more offspring than others.

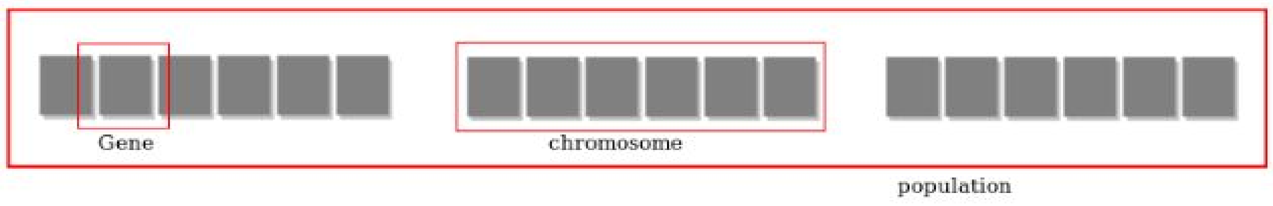
3. Genes from “fittest” parent propagate throughout the generation, that is sometimes parents create offspring which is better than either parent.

4. Thus each successive generation is more suited for their environment.

Search space:

The population of individuals is maintained within the search space. Each individual represents a solution in the search space for a given problem. Each individual is coded as a finite length vector (analogous to chromosome) of components. These variable components are analogous to Genes. Thus a chromosome (individual) is composed of several genes (variable components).

Fitness Score:



A Fitness Score is given to each individual which shows the ability of an individual to "compete". The individual having optimal fitness score (or near optimal) are sought. The GAs maintains the population of n individuals (chromosome/solutions) along with their fitness scores. The individuals having better fitness scores are given more chance to reproduce than others. The individuals with better fitness scores are selected who mate and produce better offspring by combining chromosomes of parents. The population size is static so the room has to be created for new arrivals. So, some individuals die and get replaced by new arrivals eventually creating new generation when all the mating opportunity of the old population is exhausted. It is hoped that over successive generations better solutions will arrive while least fit die.

Each new generation has on average more "better genes" than the individual(solution) of previous generations.Thus each new generation shave better"partial solutions" than previous generations. Once the offsprings produced having no significant difference than offspring produced by previous populations, the population is converged. The algorithm is said to be converted to a set of solutions for the problem.

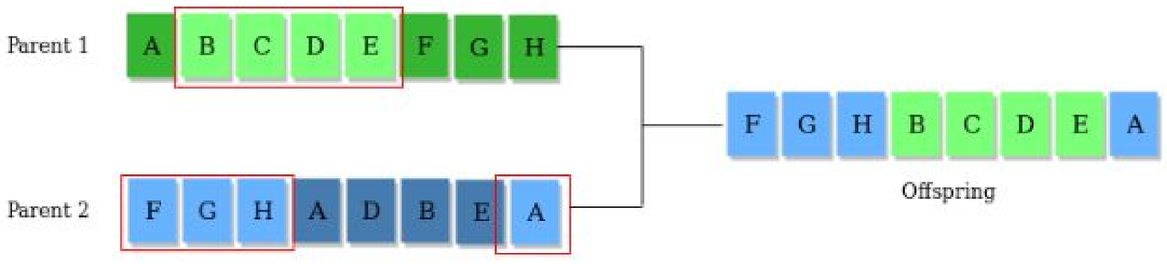
Operators of Genetic Algorithms:

Once the initial generation is created, the algorithm evolve the generation using the following operators –

1)**Selection Operator**: The idea is to give preference to the individuals with good fitness scores and allow them to pass their genes to the successive generations.

2)**Crossover Operator**: This represents mating between individuals. Two individuals are selected using selection operator and crossover sites are chosen randomly. Then the genes at these crossover sites are exchanged thus creating a completely new individual (offspring).

For example :



The whole algorithm can be summarized as:

1) Randomly initialize populations p

2) Determine the fitness of the population

3) Until convergence repeat:

   a) Select parents from population

   b) Crossover and generate a new population

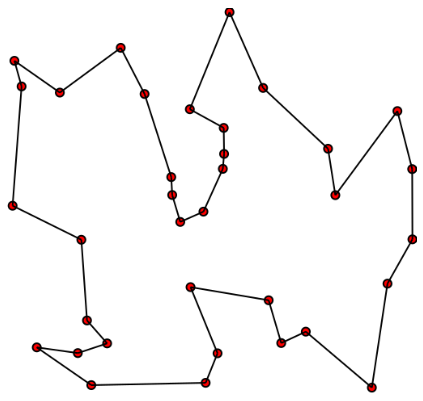
   c) Perform mutation on new population

   d) Calculate fitness for the new population

The Problem

We'll be using a GA to find a solution to the traveling salesman problem (TSP). The TSP is described as follows:

"Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city and returns to the origin city?"



Given this, there are two important rules to keep in mind:

- Each city needs to be visited exactly one time
- We must return to the starting city, so our total distance needs to be calculated accordingly

The Approach

Gene: a city (represented as (x, y) coordinates)

Individual (aka "chromosome"): a single route satisfying the conditions above

Population: a collection of possible routes (i.e., collection of individuals)

Parents: two routes that are combined to create a new route

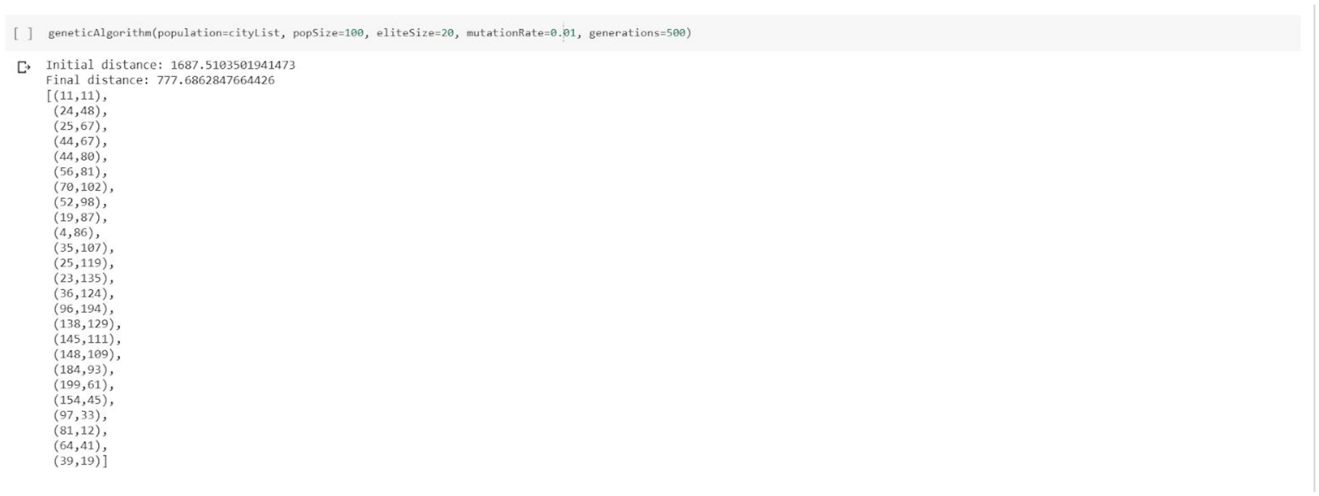
Mating pool: a collection of parents that are used to create our next population (thus creating the next generation of routes)

Fitness: a function that tells us how good each route is (in our case, how short the distance is)

Mutation: a way to introduce variation in our population by randomly swapping two cities in a route

Elitism: a way to carry the best individuals into the next generation

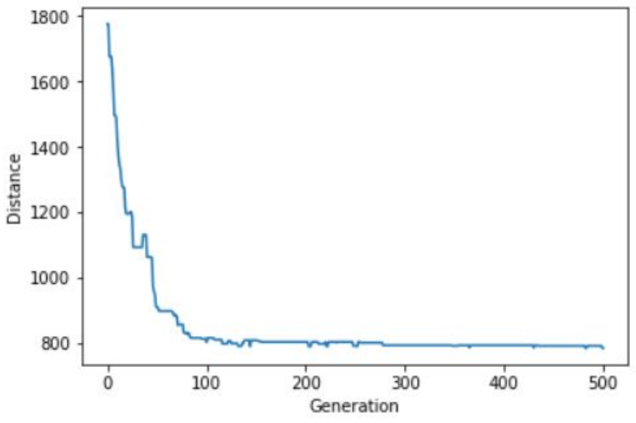
OUTPUT:



```
[ ]  geneticAlgorithm(population=cityList, popSize=100, eliteSize=20, mutationRate=0.01, generations=500)

     Initial distance: 1687.5103501941473
     Final distance: 777.6862847664426
     [(11,11),
      (24,48),
      (25,67),
      (44,67),
      (44,80),
      (56,81),
      (70,102),
      (52,98),
      (19,87),
      (4,86),
      (35,107),
      (25,119),
      (23,135),
      (36,124),
      (96,194),
      (138,129),
      (145,111),
      (148,109),
      (184,93),
      (199,61),
      (154,45),
      (97,33),
      (81,12),
      (64,41),
      (39,19)]
```

```
[ ]  geneticAlgorithmPlot(population=cityList, popSize=100, eliteSize=20, mutationRate=0.01, generations=500)
```



CONCLUSION:

Hence in this project we understood the working of Genetic algorithm used on TSP.